A NEW PERSPECTIVE IN UNDERSTANDING OF ADAM-TYPE ALGORITHMS AND BEYOND

Anonymous authors
Paper under double-blind review

ABSTRACT

First-order adaptive optimization algorithms such as Adam play an important role in modern deep learning due to their super fast convergence speed in solving large scale optimization problems. However, Adam's non-convergence behavior and regrettable generalization ability make it fall into a love-hate relationship to deep learning community. Previous studies on Adam and its variants (refer as Adam-Type algorithms) mainly rely on theoretical regret bound analysis, which overlook the natural characteristic reside in such algorithms and limit our thinking. In this paper, we aim at seeking a different interpretation of Adam-Type algorithms so that we can intuitively comprehend and improve them. The way we chose is based on a traditional online convex optimization algorithm scheme known as mirror descent method. By bridging Adam and mirror descent, we receive a clear map of the functionality of each part in Adam. In addition, this new angle brings us a new insight on identifying the non-convergence issue of Adam. Moreover, we provide new variant of Adam-Type algorithm, namely AdamAL which can naturally mitigate the non-convergence issue of Adam and improve its performance. We further conduct experiments on various popular deep learning tasks and models, and the results are quite promising.

1 INTRODUCTION

In recent years, first-order optimization algorithms with adaptive learning rate have become the dominant method to train deep neuron networks because these methods show extraordinary power on solving large-scale machine learning optimization problems. By cooperating with first-order information, adaptive methods iteratively update parameters by moving them to the direction of the negative gradient of the cost function with non-fixed learning rate. The first algorithm in this line of research can be dated back to (McMahan & Streeter, 2010), where they demonstrate that the convergence rates can often be dramatically improved through the use of preconditioning. Then AdaGrad (Duchi et al., 2011) provides first practical adaptive algorithm with theoretical guarantee based on (Zinkevich, 2003) regret analysis. Although AdaGrad achieves significant improvement on sparse settings, the rapid decay of the learning rate limits it usage. This is because AdaGrad use the past gradient accumulation as adaptive learning rate. To address this, several variants of AdaGrad, such as RMSProp (Hinton et al., 2012), AdaDelta (Zeiler, 2012), Adma Kingma & Ba (2014) have been proposed to mitigate the rapid decay of the learning rate. In particular, Adam use exponential moving average (EMA) to obtain smooth rate.

Denote $g_t \in \mathcal{R}^d$ as the gradient of generic optimization problem f with respect to its parameters $x \in \mathcal{R}^d$ at iteration t, then the generic updaing rule of adaptive methods can be express as follows (Reddi et al., 2019):

$$x_{t+1} = x_t - \frac{\alpha_t}{\sqrt{v_t}} \odot m_t \tag{1}$$

where $\odot$ denotes the entry-wise or Hadamard product. In equation above, $m_t = \vartheta(g_1, \cdots, g_t)$ is a function relates to the historical gradients; $v_t = v(g_1, \cdots, g_t)$ is a n-dimension vector with non-negative entry; α_t is the base learning rate; For Adam, in particular, the m_t and v_t are computed by EMA of gradient, with coefficient β_1 and β_2 where

$$m_t = (1-\beta_1)\sum_{i=1}^{i=t} \beta_1^{t-i} g_i \quad \text{and} \quad v_t = (1-\beta_2)\sum_{i=1}^{i=t} \beta_2^{t-i} g_i^2 \tag{2}$$

Adam, the most popular adaptive method, has been widely adopted by deep learning community. The root cause of the fast convergence of Adam and its variants in convex or non-convex optimization problems remains an open question (Chen et al., 2018). In addition, the generalization ability and out-of-sample behavior of Adam and all other adaptive methods are even worse than traditional non-adaptive counterparts such as vanilla stochastic gradient descent (Vanilla SGD) (Wilson et al., 2017). In order to understand the insight behind Adam algorithm and close the generalization gap, several Adam-Type algorithms have been proposed including (Reddi et al., 2019; Luo et al., 2019; Zhou et al., 2018; Balles & Hennig, 2017; Liu et al., 2019). Although they propose many different kinds of viewpoints in understanding the performance of Adam and demonstrate a series of correction methods to improve Adam, we think the behavior of Adam-Type algorithms is still unclear. For example, one common thinking about about m_t and v_t in Adam is first and second moments of unbiased estimator g_t, however, why this second moments can be used as adaptive learning rate? Also, another commonly asked question is where the Adam adopts such a fast convergence speed?

In this paper, in order to answer the questions mentioned above and have a deep comprehension on Adam-Type algorithms, we provide a new insight into adaptive learning rate methods, which brings a new perspective on identifying the non-convergence issue of Adam. In the previous work, the behavior analysis of Adam is based on Kingma & Ba (2014) framework, which limits our understanding. In fact, the adaptive first-order methods has a long history. Adaptive learning rate was first mentioned in Streeter & McMahan (2010); McMahan & Streeter (2010) but it is highly related to adaptive regularization of follow-the-proximally-regularized-leader (FTPRL). We notice that the intrinsic design of Adam can be related to the traditional mirror descent method (Xiao, 2010). The more detail and our motivation can be found in next section. We summarize our contribution in two folds:

1. We provide a new perspective in understanding the non-convergence behavior of Adam-Type algorithms based on mirror descent approach.
2. Based on our observation, we identify potential fault in Adam-Type algorithms and we provide a new Adam variant algorithm, AdamAL.

2 Preliminaries and motivations

Notations Given a vector $x \in \mathcal{R}^d$ we denote its i-th entry by x_i; We use $||x||$ to denote its l_2 norm; for a vector x_t in the t-th iteration, the i-th coordinate of x_t is denoted as $x_{t,i}$. Given two vectors $x, y \in \mathcal{R}$, we use $\langle x, y \rangle$ to denote their inner product, $x \odot y$ to denote element-wise product, $\frac{x}{y}$ to denote entry-wise division, the $\max(x, y)$ to denote entry-wise maximum and $\min(x, y)$ to denote entry-wise minimum. We use $\mathcal{S}_+$ to denote the set of all positive definite matrices M. We use $M^{\frac{1}{2}}$ to denote $\sqrt{M}$.

2.1 Preliminaries

Nonlinear projected subgradient methods and mirror descent algorithm Iterative gradient descent (GD) scheme, which can be traced back to (Cauchy, 1847), is the simplest strategy to minimize convex optimization problems. It was further developed as Zinkevich Online Greedy Subgradient Project (OGSP) (Zinkevich, 2003), which can be considered as a variation of project gradient descent (PGD) algorithm with following updating rules:

$$x_{t+1} = P_{\mathcal{X}}(x_t - \eta f^{'}(x_t)) \text{ (standard PGD)}, \quad x_{t+1} = P^A_{\mathcal{X}}(x_t - \eta_t g_t) \text{ (Zinkevich OGSP)} \tag{3}$$

where $P^A_{\mathcal{X}}$ is a distance-based projector denoting the projection of a point y onto $\mathcal{X}$ by $P^A_{\mathcal{X}}(y) = \arg\min_{x \in \mathcal{X}} ||x - y||_A$, $|| \cdot ||_A = \langle x, Ax \rangle$, and $g_t \in \partial f_t(x_t) = f^{'}(x_t)$ is the subgradient of the objective function $f_t(\cdot)$. The major issue with applying PGD is that PGD only works in Hilbert space $\mathcal{H}$ and it cannot be extended to more general situation (in modern machine learning) of optimization in some Banach space $\mathcal{B}$ where the Euclidean norms cannot be computed. To this end, the mirror descent algorithm (MDA) introduced by (Nemirovsky & Yudin, 1983) overcomes such infeasibility by using the linearity on dual vector space $\mathcal{B}^*$ and a carefully designed mirror map (Bubeck, 2014). MDA has following updating schemes:

$$x_{t+1} = \arg\min_{x \in \mathcal{X}} \{ \langle f^{'}(x_t), x \rangle + \frac{1}{\eta_t} D_\psi(\cdot) \} \tag{4}$$

By replacing the Euclidean quadratic norms in Equation. 3 with more general distance-liked settings such as Bregman distance function $D_\psi(\cdot)$ define on $\psi(\cdot)$, the equivalence of PGD algorithm and MDA has been proved by Beck & Teboulle (2003). For example, the simplest version of MDA is that: taking $\psi(x) = \frac{1}{2}||x||_2^2$, and $D_\psi(x, x_t) = \frac{1}{2}||x - x_t||_2^2$ then plugging into Equation. 4, we have:

$$x_{t+1} = \arg\min_{x \in \mathcal{X}} \{\langle g_t, x\rangle + \frac{1}{2\eta_t}||x - x_t||_2^2\} \tag{5}$$

Update scheme in the above equation. 5 is also known as **proximal point algorithm** (Rockafellar, 1976). The equivalence of Equation. 5 and PGD can be achieved by taking the derivative of our target function on x, and rearranging the formula:

$$x_{t+1} = x^* = x_t - \eta_t g_t = P_{\mathcal{X}}^I(x_t - \eta_t g_t) \tag{6}$$

Now, we see the last two expressions in Equation. 6 are well-known subgradient descent update. We restate the proposition 3.2 in Beck & Teboulle (2003) as:

Proposition 1. *Assume $\mathcal{X}$ is a closed convex subset in $\mathcal{R}$ with non-empty interior, and objective function $f : \mathcal{X} \to \mathcal{R}$ is a convex and Lipschitz function. Suppose the optimal set of x denoted by $\mathcal{X}^*$ is non-empty, we can compute the subgradient of f at x as $g \in \partial f(x)$. For a convex mirror mapping function $\psi : \mathcal{X} \to \mathcal{R}$ with conjugate function ψ^* defined by $\psi^*(y) = \max_{x \in \mathcal{X}}\{\langle x, y\rangle - \psi(x)\}$. Then the sequence $\{x_t\} \subseteq \mathcal{X}$ generated by MDA is equivalent to the sequence generated by PGD.*

We state the equivalence of PGD algorithm and MDA, particularly, the general gradient descent (DG or SGD) can be directly derived from the Equation. 6.

Follow the proximally-reqularized leader (FTPRL) FTPRL is introduced by McMahan & Streeter (2010) belongs to the family of follow-the-regularized-leader (FTRL) algorithm such as Regularized Dual Averaging (Xiao, 2010). In general, FTRL-Type has following update rule:

$$x_{t+1} = \arg\min_{x \in \mathcal{X}} \{(\sum_{\tau=1}^{t} f_\tau^{'}(x_\tau)) \cdot x + R_{1:t}(x) \tag{7}$$

where the subgradient of objective function $f_\tau^{'}(x_\tau)$ is approximated by the gradient at x_τ and $R_{1:t}(x)$ is defined as regularization. Particularly, the formal FTPRL is:

$$x_{t+1} = \arg\min_{x \in \mathcal{X}} \{g_{1:t} \cdot x + \phi_{1:t} \cdot x + \Psi(x) + \frac{1}{2}\sum_{\tau=1}^{t} ||Q_\tau^{\frac{1}{2}}(x - x_\tau)||_2^2\} \tag{8}$$

with $\phi_{1:t} \cdot x + \Psi(x)$ can be considered as non-smooth composite term which is orthogonal to our paper, more detail can be found in (McMahan, 2010b). The last term in above equation is stabilizing regularization that ensure low regret. It is also worth mentioning that the Q_τ can be regarded as **generalized learning rate** which plays crucial role in this paper. As we can see, FTPRL appears quite different from MDA stated in Equation. 5, however, in McMahan (2010a;b) they show that in the case of selecting quadratic stabilizing regularization, the FTPRL and generalized MDA only has differences in parameter centering. In fact, MDA illustrates in Equation. 5 regularizing the parameter to be close to the origin, on contrast, FTPRL is regularizing the parameter at current feasible point. No surprising, McMahan (2010a) propose the equivalence proof of FTPRL and a variation algorithm of the MD as follow.

Proposition 2. *Let R_t be a sequence of differentiable convex functions ($\nabla R_t(0) = 0$), and let Ψ be an arbitrary convex function. Define the proximal-MDA with updating rule:*

$$x_{t+1} = \arg\min_{x \in \mathcal{X}} \{\langle g_t(x_t), x\rangle + \Psi(x) + D_{R_{1:t}}(\cdot)\} \tag{9}$$

where the Bregman distance function D_{R_t} with respect to R_t where $R_t(x) = R_t(x - x_t)$. And applying FTPRL to the same objective function, with:

$$x_{t+1} = \arg\min_{x \in \mathcal{X}} \{g_{1:t} \cdot x + \phi_{1:t} \cdot x + \Psi(x) + R_{1:t}\} \tag{10}$$

when $\phi_t \in \partial\Psi$, such that $g_{1:t} + \phi_{1:t} + \nabla R_{1:t}(x_t) = 0$. Then the two above update scheme are equivalent.

At this moment, we state a series of equivalence proposition from FTPRL to proximal-MDA (variation of MDA) and MDA to PGD. Now, we can construct the equivalence proof of FTPRL and PGD properly. Although the direct proof of equivalence between FTPRL and PGD is provided unofficially in McMahan (2010a), we would like to elaborate the intuition behind each algorithm and deduce our perspective to understanding the-sate-of-art algorithm such as adaptive method.

2.2 Motivation

Non-Adam-Type and Adam-Type algorithms to mirror descent (FTPRL) Non-Adam-Type online gradient descent algorithm including SGD, SGD with momentum, Polyak's HB and Nesterov's accelerated gradient method can be easily understood as first order optimization with different momentum function. And, interestingly, most of them have physical interpretations. However, as shown in Table, Adam-Type algorithms do not rely on the non-increasing learning rate when iteratively updating their parameters, instead, they perform adaptively learning rate element-wise on parameters according to the first order information. This leads to the most mystery part in the Adam-Type algorithm where the adaptively update is represented as $-\frac{\eta_0}{\sqrt{v_t}} \odot m_t$. One commonly interpretation of this updating representation is regarding $-\frac{\eta_0}{\sqrt{v_t}}$ as adaptive learning rate, and regarding m_t as general first order gradient. However, we can not treat Adam-Type algorithm in this way because the first order information g_t resides in both m_t and v_t and we are unable to simply decorrelate them as learning rate scheme. Another possible interpretation of such updating is related to the Newton's second order method, but there is no free lunch for expressing in this way. We will discuss it later.

In order to dissect Adam-Type algorithm, we recall the FTPRL algorithm mentioned in the previous section. A natural question raise: can we interpret Adam-Type algorithm as a variant of Mirror Descent method? Before answer the question, let us explain why we present Adam-Type algorithm as MD is beneficial? We summary in the following:

1. **Implicit updates:** This concept was first derived by Kivinen & Warmuth (1997) and later pointed by Kulis & Bartlett (2010). It refers that the without using explicit first-order update rules to efficiently compute the parameters. The mirror descent algorithms, in fact, adopt implicit update rules very well, therefore, the explicit update will roll into some regularized-liked terms which can elaborate more insights. In contrast, like Adam, the learning rate is explicitly defined as $\frac{\eta_0}{\sqrt{v_t}}$ which is hard to identify from intuition unless relying on the theoretically proofs.
2. **First-order information dissection** The entire adaptive update involves in Hadamard product and division. Both numerator and denominator are highly related to the construction function respect to the first-order gradient g. The underlying relation between two functions m_t and v_t can not be decorrelated. However, as show in Table, we represent Adam-Type algorithms in MD way so that we can separate the numerator and denominator as simple additive scheme where the hard-understanding division disappear and meanwhile, the alignment of m_t and v_t gone.
3. **Equivalence guarantee** The original adaptive method AdaGrad is build upon non-linear subgradient projection, and the Adam actually inherit such design indicate by their regret proof. To re-represent Adam-Type algorithm as FTPRL style, we do require the theoretically equivalence guarantee so that we can transform safely. As we discuss in previous (Proposition 1& 2), we successfully build such bridge in between two types algorithm.

In general, these first-order algorithms can be written in FTPRL style:

$$x_{t+1} = \arg\min_{x\in\mathcal{X}}\{\rho\underbrace{(g_{1:t}+C_{1:t})\cdot x}_{A} + \underbrace{\Psi(x)}_{B} + \underbrace{\frac{1}{2}\sum_{\tau=1}^{t}||Q_\tau^{\frac{1}{2}}(x-x_\tau)||_2^2\}}_{C} \tag{11}$$

The understanding of above representation is highly related to our analysis on Adam-Type algorithms. Term A has two parts, the first part $g_{1:t}\cdot x$ is an approximation to $f_{1:t}$ based on the gradient; the second part $C_{1:t}\cdot x$ refers as *first-order momentum correction* or *fault tolerant* in this literature. Term B is similar to the setting of FTPRL, but we usually consider it as l_2 regularization. In addition, term C stabilizing regularization plays very crucial role in this transformation because (1) the implicit updates from x_t to x_{t+1} happens in this term; (2) the Q_τ reside in norms can be regarded as generalized learning rate or even more complicated format; (3) the rule of parameter centering to current feasible solution is figured. Finally, the leading ρ is a *balancing coefficient*, aims at controlling the tendency of minimization between term A and C. Smaller ρ value will guide the minimization process relies more on term C, otherwise term A will dominate the minimization.

The simplest format transformation from Vanilla SGD to MD is illustrate in Equation. 5. Now, we rewrite it as FTPRL style according to Eqution. 8 (for the sake of simplicity, we do not consider the Ψ temporarily, we will discuss it later):

$$x_{t+1} = \arg\min_{x\in\mathcal{X}}\{g_{1:t}\cdot x + \frac{1}{2}\sum_{\tau=1}^{t}\sigma_\tau||x-x_\tau||_2^2\} \tag{12}$$

where the $\rho = 1$, $C_{1:t}$ is zero, and Q_τ sets to be $Q_\tau = \sigma_\tau^2\boldsymbol{I}$. If Vanilla SGD with learning rate $\eta_t = \frac{1}{t}$, the $\sum_{\tau=1}^{t}\sigma_\tau = t$. If it with fixed learning rate $\eta_t = \frac{1}{k}$, then $\sum_{\tau=1}^{t}\sigma_\tau = k$. To be mentioned here, fixed learning rate SGD is kind of special, it can be easily explained as Equation. 5, however, in FTPRL format (Equation. 12), we need to be more careful.

Mirror descent liked Vanilla SGD with non-increasing learning rate show the great insight of understanding current algorithms. First, we are always looking for the next step x_{t+1} in the opposite direction of the current gradient because $\cos(\pi) = -1$ minimize Equation. 12. Second, x_{t+1} is being bounded in the a region centered at previous step x_t. This is due to the fact that we do not want the new solution be far away from the current feasible solution, otherwise, may cause slow convergence, McMahan (2010b) confirm this view. Third, $\sum_{\tau=1}^{t}\sigma_\tau$ should be non-decreasing alone the time, similar, when σ_τ get smaller, the bounding region expand reducing the convergence speed.

3 New Perspective on Adam

In this section, we deliver a new perspective on Adam-Type algorithm from the Mirror Descent point of view. In this lecture, we mainly focus on Adam to demonstrate our analysis, for the other variants, they have similar results.

3.1 Adam on Mirror Descent

According to the Table, the subgradient projected Adam can be written as follows (entry-wise). In order to make a clear comparison, we also show SGDM alongside.

$$\begin{aligned}
&\text{Adam: } x_{t+1,i} = \arg\min_{x\in\mathcal{X}}\{(g_{1:t,i} - \sum_{\tau=1}^{\tau=t,j=t}\beta_1^{j+1-i}g_{\tau,i})\cdot x_{,i} + \frac{1}{2}\sum_{\tau=1}^{t}\sigma_{\tau,i}||x_{,i}-x_{\tau,i}||_2^2\} \\
&\text{SGDM: } x_{t+1,i} = \arg\min_{x\in\mathcal{X}}\{\frac{1}{1-\beta_1}(g_{1:t,i} - \sum_{\tau=1}^{\tau=t,j=t}\beta_1^{j+1-i}g_{\tau,i})\cdot x_{,i} + \frac{1}{2}\sum_{\tau=1}^{t}\sigma_\tau^*||x_{,i}-x_{\tau,i}||_2^2\}
\end{aligned} \tag{13}$$

where both $\beta_1 \leq 1$ (commonly chose $\beta_1 = 0.9$), the $\sum_{\tau=1}^{t}\sigma_{\tau,i} = \sqrt{(1-\beta_2)\sum_{\tau=1}^{\tau=t}\beta_2^{t-\tau}g_{\tau,i}^2}$ in Adam settings with $\beta_2 = 0.999$. In SGDM, $\sum_{\tau=1}^{t}\sigma_\tau^*$ performs differently, (1) SGDM is non-adaptive method, the τ^* applies to all entry on the x; (2) $\sum_{\tau=1}^{t}\sigma_\sigma^* = t$ if we have learning rate $\eta_t = \frac{1}{t}$, it also means $\sigma^* = 1$ for all $\tau \in T$. First, let us recall Vanilla SGD and compare it with SGDM:

Corollary 1. *Compare to Vanilla SGD, SGDM employs first-order momentum correction defined in Section. 2.2 to correct the possible wrong direction prediction, making a smooth optimization trajectory. (This is a well known truth. We verify it by experiments show in Appendix.)*

Besides, we also notice that Adam has $\rho = 1$ while SGDM has $\rho = \frac{1}{1-\beta_1} = 10$. Recall the previous section, by definition of ρ, we have following corollary:

Corollary 2. *Compare to Vanilla SGD or Adam, SGDM has larger ρ value indicates that the SGDM is* ***more sensitive on the value change of loss function****.*

To explain this, we know one of the drawbacks of SGDM is that SGDM is prone to oscillation around the optimal point, because SGDM has relative weak bound in proximal term, and is very sensitive on small change of loss, that is, ρ factor amplifies this loss change up to tenfold.

Now, back to the Adam, before we move forward, we define some terms in Equation. 13.Adam,

Proximal Searching Region refer as $D = ||x - x_\tau||_2^2$, this Euclidean quadratic norm reflects the geometry of given constraints feasible set $\mathcal{X}$. We can also treat it as a regularization process. This region is inversely proportional ($D \propto \frac{1}{B}$) to the Regularization Budget defined below.

Regularization Budget refer as $B = \sum_{\tau=1}^{t} \sigma_{\tau,i}$, it indicates total "weight" we can distribute to the Proximal Searching Region. More weight it has been given will lead to a stronger regularization in bounding, and a small searchable space centering at x_τ.

In Hoffer et al. (2017), they define an ultra slow diffusion phenomenon when they evaluate the distance from current weight to initialization weight point with $||x_t - x_0||_2^2 \sim \log t$. Interestingly, this result is entirely consistent with our Proximal Searching Region analysis, because for any $t \in T$, we have $||x_{t+1} - x_t|| \sim (logt)^{'} = \frac{1}{t}$. Now, we summarise our result as follow:

1. **Hyper-parameter β_1:** β_1 exponential smooth **eliminates** the presence of **imbalance** in between the goal of minimizing loss function and the constraints of searching in proximal region. In other words, Adam treats both conditions fairly.
2. **First-order momentum:** the usage of m_t leads to a smooth optimization trajectory which avoid the sharp twist such as SGD. It can benefit Adam if searching in wrong direction, the momentum correction C_τ can compensate the party of penalty directly from the loss function.
3. **Adam-Type algorithm:** Adam-Type algorithms such as AdamG (this paper), AMSGrad AdaBound, AdaShift, NosAda, etc can be regarded as making a correction on one of regularization term (most on $||Q_\tau^{\frac{1}{2}}||x - x_\tau||_2^2$).
4. **Learning rate:** mirror descent corporate with implicit update makes learning rate (step size) act as a scalar factor of regularization.

By transferring the Adam to MD-liked method, we successfully disassemble the Adam updating $-\frac{m_t}{\sqrt{v_t}}$ into two additive terms and identify their functionality separately. We mainly focus on the m_t part in this section, we will move our eye on $\frac{1}{\sqrt{v_t}}$ in next section.

3.2 Adam V_t and the non-convergence of Adam

A commonly thinking of Adam's adaptive learning rate $\frac{1}{\sqrt{v_t}}$ will treat v_t as second moment approximation. However, in our perspective, we regard it as **adaptive regularize scalar** and performs implicit update by replacing the explicit learning rate. Two adaptive regularize from AdaGrad and Adam show in below:

$$\text{Adam: } \sum_{\tau=1}^{t} \sigma_{\tau,i} = \sqrt{(1-\beta_2)\sum_{\tau=1}^{\tau=t} \beta_2^{t-\tau} g_{\tau,i}^2} \quad \text{AdaGrad: } \sum_{\tau=1}^{t} \sigma_{\tau,i} = \sqrt{\sum_{\tau=1}^{\tau=t} g_{\tau,i}^2} \tag{14}$$

By guiding with this intuition and our regularization budget definition, we know can conclude that

1. Strictly speaking, non-Adam-Type algorithm such as SGD(M) has only proximal searching region **center at** x_t, however, Adam-Type algorithms achieve globally stabilizing regulations via proximal searching region through $\{x_1, x_2, \cdots, x_{t-1}, x_t\}$. Therefore, in our settings, SGD has σ function with a constant value where $\{\sigma_\tau = k | \tau = t\}$. A *special case* is when $\eta_t = \frac{1}{t}$ where the SDG has a descending step size $\frac{1}{t}$, we notice that $\sum_{\tau=1}^{t} \sigma_\tau = t$ and $\sigma_t = 1$. In fact, in our theory, we can regard SGD with decreasing learning rate as a adaptive regularization with the respect to training iterations.
2. Adam-Type algorithms, in contrast, have stronger bounding constraints with each proximal term $||x - x_\tau||_2^2$. With the training iterations increasing, in order to retain the the similar regularization strength, the natural way is increasing the regularization budget $\sum_{\tau=1}^{t} \sigma_\tau$ such that $B_t \geq B_{t-1}$. We have the following comments for Adam:

2.1 Pro: Non-decreasing regularization budget B_t can benefit for Adam, however, unbounded budget cause the infinitely small proximal searching region, on the surface, training will stop without parameters update. Adam over come this problem wisely, exponential moving average (EMA) preforms as a sliding window, in other words, regularization budget is bounded.

2.2 Con: EMA, on the one hand controls the regularization budget in a range, but it fails to satisfy the primary requirement that regularization budget B_t should be in non-decreasing manner. Lot of previous works point out this issue such as (Reddi et al., 2019; Luo et al., 2019; Chen et al., 2018). However, the way the identify such problem is not natural, for example, the objective function with periodicity gradient rarely seen in real scenario. Our way seems more easy to access.

The non-convergence of Adam This issue was first identified by Reddi et al. (2019), which points out that the key issue in the convergence of Adam lies in the quantity

$$\Gamma_t = (\frac{\sqrt{v_t}}{\eta_t} - \frac{\sqrt{v_{t-1}}}{\eta_{t-1}}) \tag{15}$$

which assumes to be a non-negative value, but in training, this assumption dose not always hold in Adam. Reddi et al. (2019) construct an objective function with periodicity gradient to illustrate the non-convergence of Adam which is hard to follow. And (Luo et al., 2019) using the similar way but conduct a heuristic experiment shows that the Adam will generate extreme learning rates (also extreme v_t value) can effect the convergence. Fundamentally, they are identifying the same problem in different expressions. In Reddi et al. (2019) non-convergence Theorem.1 the repeated occurrence gradient $-C$ is, in fact, the extreme learning rates. In our analysis, the emerge of extreme learning rates is due to decreasing regularization budget $B_t - B_{t-1} < 0$ which equivalent to $\Gamma < 0$, the negative regularization coefficient $\sigma_\tau < 0$ makes the corresponding proximal searching region $||x - x_\tau||_2^2$ to be infinite large when minimizing the Equation. 13.Adam. Again, the parameters behavior in out of the control manner for example $||x - x_\tau||_2^2 \to +\infty$. To this end, Reddi et al. (2019) propose a very intuitive modification on Adam where

$$v_t = \max\{v_t, v_{t-1}\} \tag{16}$$

Although this setting solve the decreasing regularization budget issue, it still remain the problem so called nonalignment projection, we state this problem in next section.

3.3 NON-ALIGNMENT PROJECTION AND ADAMAL

AMSGrad (Reddi et al., 2019) using Equation. 16 to ensure the $\Gamma_t \geq 0$ for all $t \in T$. They derive it mainly from an unrealistic objective function. Does it really solve the problem or dose this design violate the intuition of Adam-Type algorithm? We conduct the experiment to illustrate the fundamental problem of AMSGrad (or its variant AdaShift). We call this problem as **non-alignment projection**. To illustrate the non-alignment projection problem, we trace a series of entries generated by AMSGrad and sample them from $v_{t,i:j}$, our goal is counting the total times of that entry being swap with $v_{t-1,i:j}$ and meanwhile, we record the interval between two swaps. We employ a heatmap to visualize this result. As show in figure.1, we can find that (1) different entry has very different swapping counts; (2) the swapping intervals are nonuniform. Note here sampling $v_{t,i}$ from different neuron network layers and different batch size have different results, but we demonstrate the presence of such problem. To be more specifically, we present a simple one-step AMSGrad swapping at iteration t and figure out the ill-condition problem.

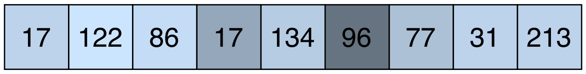
Suppose at iteration t we have corresponding v_t, then next step. we have $v_{t+1} = \beta_2 v_t + (1-\beta_2) g_{t+1}^2$, at this moment, according to AMSGrad, we evaluate $v_{t+1} = \max\{v_{t+1}, v_t\}$, if the swap happens, we will receive $v_{t+1} = v_t$. We likely to use this v_{t+1} for next step updating namely $v_{t+2} = \beta_2 v_{t+1} + (1-\beta_2) g_{t+2}^2$. However, the real value of v_{t+2} should be

$$v_{t+2} = \beta_2 v_{t+1} + (1-\beta_2) g_{t+2} = \beta_2^2 v_t + (1-\beta_2)\beta_2 g_{t+1}^2 + (1-\beta_2) g_{t+2}^2 \tag{17}$$

and AMSGrad will gives us

$$v_{t+2}^{ams} = \beta_2 v_t + (1-\beta_2) g_{t+2}^2$$

The difference is quite obvious show in here, but another question is Why AMSGrad still working? AMSGrad can still work is because we choose very big β_2 value as 0.999 which makes the



17	122	86	17	134	96	77	31	213

Figure 1: The number indicates the total swapping by AMSGrad at $v_{t,i}$. The darker heatmap colors reveal the lower average swapping interval.

$v_{t+2}^{ams} \approx v_{t+2}$. In fact, iterative method will accumulate this small error into each step and as a consequence, the non-alignment of v_t will lead the model to find a suspicious local optimal. Another non-alignment refers to m_t and v_k, AMSGrad updates its i^{th} parameter by

$$x_{t+1,i} = x_{t,i} - \frac{m_{t,i}}{\sqrt{v_{k,i}}} \quad \text{where } k \neq t$$

Recall in the Zinkevich's greedy projection, we minimize the objective loss function relies on current gradient approximation g_t then we perform the projection by projector $\mathcal{P}_{\mathcal{X}}^{-\sqrt{v_k}}$ to resolve the x_{t+1}. We see the v_k is not the actual projection matrix for x_t approximate by m_t.

To address above issue, recall the define of non-decreasing regularization budget, in Adam setting, $B_t \geq B_{t-1}$ equivalent to $v_t \geq v_{t-1}$ that is

$$v_t - v_{t-1} = (1-\beta_2)(g_t^2 - v_{t-1}) \geq 0 \Rightarrow g_t^2 - v_{t-1} \geq 0$$

Now, the solution for resolving Adam's non-convergence and non-alignment of AMSGrad is clear. That is before we evaluate v_t, we modify g_t to guarantee the non-decreasing condition of v_{t-1} to v_t. We illustrate the update detail of AdamAL in Algorithm.1. Using a similar one-step example to illustrate how AdamAL can mitigate non-alignment issue when update v_{t+2}:

Algorithm 1 AdamAL

```
Input $x \in \mathcal{F}$, initial step size $\alpha$, $\beta_1, \beta_2$,
set $m_t = 0$, $v_t = 0$
for t = 1 to T do
    $g_t = \nabla f_t(x_t)$
    $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$
    $\hat{g}_t = \max\{g_t^2, v_{t-1}\}$
    $v_t = \beta_1 v_{t-1} + (1-\beta_1)\hat{g}_t$
    $x_{t+1} = \mathcal{P}_{\mathcal{X}}^{-\sqrt{v_t}}(x_t - \frac{\alpha}{\sqrt{v_t}} \odot m_t)$
end for
```

$$v_{t+2}^{adamal} = \beta_2 v_{t+1} + (1-\beta_2) g_{t+2} = \beta_2^2 v_t + (1-\beta_2)\beta_2 \hat{g}_{t+1}^2 + (1-\beta_2) g_{t+2}^2 \tag{18}$$

which reconstruct the v_{t+2} in correct expression. The major difference of AdamAL and AMSGrad is we **do not skip** the gradient update for v_t so that we guarantee the alignment of g_t and v_t so as m_t. We have the following bound for Adamal.

Theorem 3.1. *Suppose that the Assumption in Section.II are satisfied, β_1 is chosen such that $\beta_1 \geq \beta_{1,t}$ with $\beta_{1,t} \in (0,1)$ is non-increasing, and for some constant G, we have $||\frac{\alpha_t}{\sqrt{v_t}} m_t|| \leq G$, for all t, then the Algorithm. 1 satisfy:*

$$\begin{aligned} &E[\sum_{t=1}^{T} \langle \nabla f(x_t), \nabla f(x_t)/\sqrt{v_t}] \\ &\leq E[A_1 \sum_{t=1}^{T} ||\alpha_t g_t/\sqrt{v_t}||_2^2 + A_2 \sum_{t=1}^{T} ||\frac{\alpha_t}{\sqrt{v_t}} - \frac{\alpha_{t-1}}{\sqrt{v_{t-1}}}||_2^2 + A_3 \sum_{t=1}^{T-1} ||\frac{\alpha_t}{\sqrt{v_t}} - \frac{\alpha_{t-1}}{\sqrt{v_{t-1}}}||_2^2] + A_0 \end{aligned} \tag{19}$$

where A_0, A_1, A_2, A_3 are constants related to β_1 G.

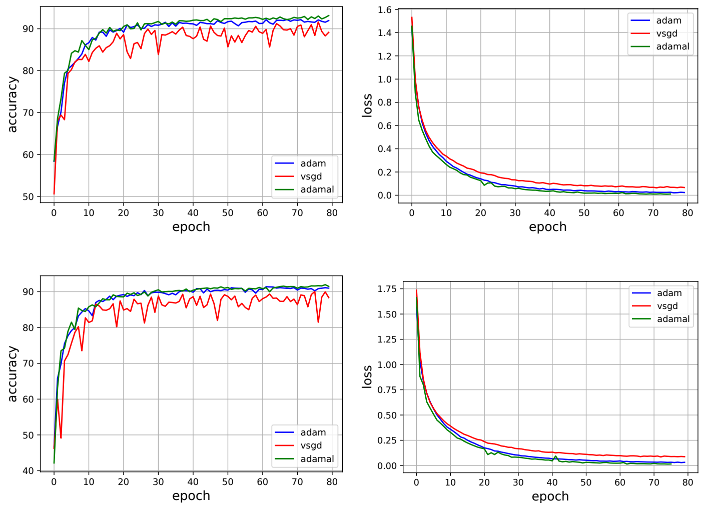


Figure 2: Upper Left: Accuracy of Cifar10 on ResNet18; Upper Right: Loss of Cifar10 on ResNet18; Bottom Left: Accuracy of Cifar10 on VGG16; Bottom right: Loss of Cifar10 on VGG16

4 EXPERIMENTS

In this section, we turn to an empirical study of different models to compare new variants with popular optimization methods including SGD(M), Adam, and AMSGrad. We focus on image classification task on CIFAR10.

From the experiment results show in Figure. 2, we notice that AdamAL outperforms Adam and AMSGrad! This result is desirable because we fix the non-alignment projection issue reside in AMSGrad. In general, from the experiments, AdamAL constanly achieve 1% accuracy gain than Adam. Notice that we only conduct our experiments with 80 epochs, this is due to that fact that we observe there is no further accuracy improvement without performing any hyperparameter tuning. If we perform hyperparameter tuning, the result show in Table.1. AdamAL can finally reach to 95% accuracy in average on test data, however, the best run of Adam still lower than AdamAL. It is also worth mentioning that AMSGrad have even worse performance than Adam due to non-alignment. We also compare the result of AdamAL using different min-batch settings, the result shows that AdamAL is also not sensitive to min-batch size.

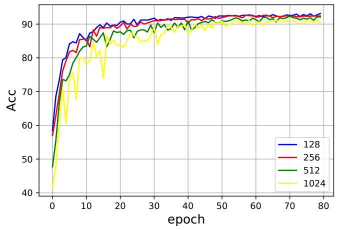


Figure 3: AdamAL in different min-batch

Algorithm	lr decay	Acc.
Adam	75, 125, 175	94.98
AMSGrad	75, 125, 175	94.60
AdamAL	75, 125, 175	95.13
VSGD	75, 125, 175	94.73

Table 1: hyperparameter tuning, learning rate halve at iteration 75, 125, 175.

5 CONCLUSION

In this paper, we present a new angle to look at the first-order method with adaptive learning rate. We decouple the Adam updating rule as an addition of two regularized terms. In this way, we can identify the intuition behind Adam-Type algorithm. Additionally, we naturally figure out the non-convergence issue resides in AMSGrad and Adam, we propose another variant of Adam algorithm to mitigate such problem.

REFERENCES

Lukas Balles and Philipp Hennig. Dissecting adam: The sign, magnitude and variance of stochastic gradients. *arXiv preprint arXiv:1705.07774*, 2017.

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Sébastien Bubeck. Theory of convex optimization for machine learning. *arXiv preprint arXiv:1405.4980*, 15, 2014.

Augustin Cauchy. Méthode générale pour la résolution des systemes déquations simultanées. *Comp. Rend. Sci. Paris*, 1847.

Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of adam-type algorithms for non-convex optimization. *arXiv preprint arXiv:1808.02941*, 2018.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *lecture*, 2012.

Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, pp. 1731–1741, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Jyrki Kivinen and Manfred K Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *information and computation*, 132(1):1–63, 1997.

Brian Kulis and Peter L Bartlett. Implicit online learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 575–582, 2010.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. *arXiv preprint arXiv:1902.09843*, 2019.

H. McMahan. Follow-the-regularized-leader and mirror descent: Equivalence theorems and implicit updates. *CoRR*, abs/1009.3240, 01 2010a.

H. Brendan McMahan. Follow-the-regularized-leader and mirror descent: Equivalence theorems and implicit updates. *CoRR*, abs/1009.3240, 2010b. URL `http://arxiv.org/abs/1009.3240`.

H Brendan McMahan and Matthew Streeter. Adaptive bound optimization for online convex optimization. *arXiv preprint arXiv:1002.4908*, 2010.

Arkadii Semenovich Nemirovsky and David Borisovich Yudin. Problem complexity and method efficiency in optimization. *J. Wiley, New York*, 1983.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

R Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.

Matthew Streeter and H Brendan McMahan. Less regret via online conditioning. *arXiv preprint arXiv:1002.4862*, 2010.

Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pp. 4148–4158, 2017.

Lin Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11(Oct):2543–2596, 2010.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Zhiming Zhou, Qingru Zhang, Guansong Lu, Hongwei Wang, Weinan Zhang, and Yong Yu. Adashift: Decorrelation and convergence of adaptive learning rate methods. *arXiv preprint arXiv:1810.00143*, 2018.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 928–936, 2003.